Comunicación de la Arquitectura de Software

Ing. Gustavo Andrés Brey
Ing. Juan Arias
Ing. Gastón Escobar

© 2005

Agenda

#	Tema	Duración
1	Concepto de Comunicación y Entendimiento de Arquitectura	30 min
2	Frameworks de Arquitectura	30 min
3	ADL	30 min
4	Guía / Metodología de Comunicación	30 min
5	Conclusión	10 min

Agenda

#	Tema
1	Concepto de Comunicación y Entendimiento de Arquitectura
2	Frameworks de Arquitectura
3	ADL
4	Guía / Metodología de Comunicación
5	Conclusión

¿Por que comunicamos?

- La Arquitectura como elemento principal para la comunicación y educación entre stakeholders
  - Siendo la primera abstracción del sistema, permite el análisis y toma de decisiones que le dan forma y estructura al proyecto. El medio de educación proviene del hecho de que la documentación de arquitectura es usada para introducir a nuevos trabajadores en el entendimiento del sistema. Estas personas bien pueden ser nuevos empleados, analistas externos o un nuevo arquitecto.
- La Arquitectura sirve como base para la evaluación de la arquitectura
  - Esta documentación debe tener la información necesaria para poder evaluar una variedad de atributos tales como seguridad, performance, usabilidad, disponibilidad y modificabilidad.
- Mecanismo de modelado y diseño para el grupo de arquitectura
  - Comunicar la arquitectura no solo es necesario para el equipo de proyecto sino tambien para el grupo de arquitectura para iterar y evolucionar la arquitectura

¿Para quién documentamos?

- Cliente
  - + Aspectos del Negocio, Instalación, Tiempos y Costo
  - – Skills. Detalles técnicos, y operacionales
- Project Manager
  - + Costos, Tiempos, Skills y Organización del Equipo.
  - – Detalles de la aplicación y operacionales.
- Infraestructura
  - + Impacto en IT, Detalles operacionales, Alocaciones y Tecnología.
  - – Aspectos del Negocio, detalles de la aplicación, interfaz de usuario.
- Analistas y Testers
  - + Aspectos del Negocio, Interfaz de Usuario, Detalles de la aplicación.
  - – Detalles técnicos, Costos.
- Programadores y Diseñadores
  - + Detalles técnicos y de la aplicación, Alocaciones, Tecnologías
  - – Costos, Impacto en el Negocio, Costos y Tiempo.

IEEE 1471-2000

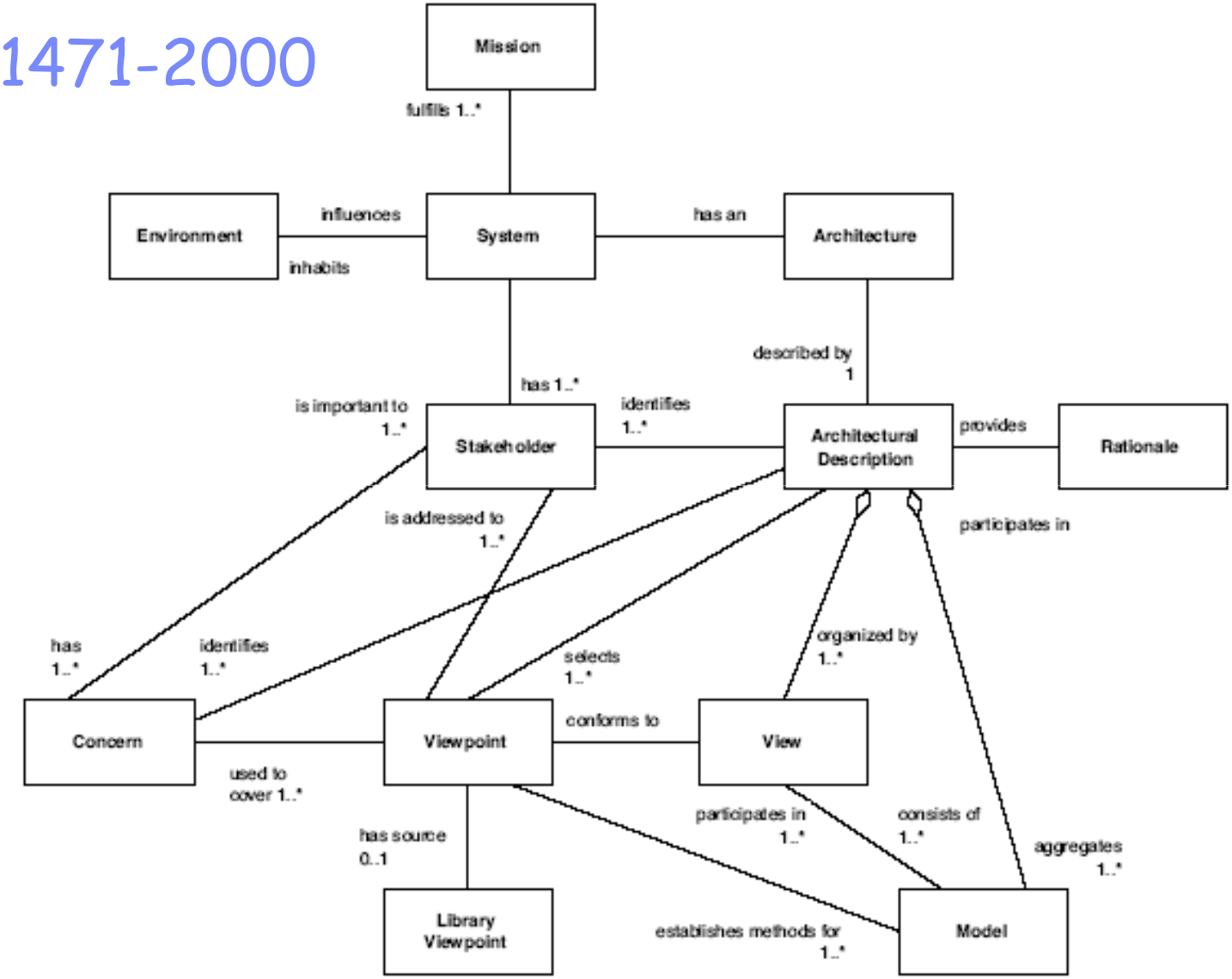
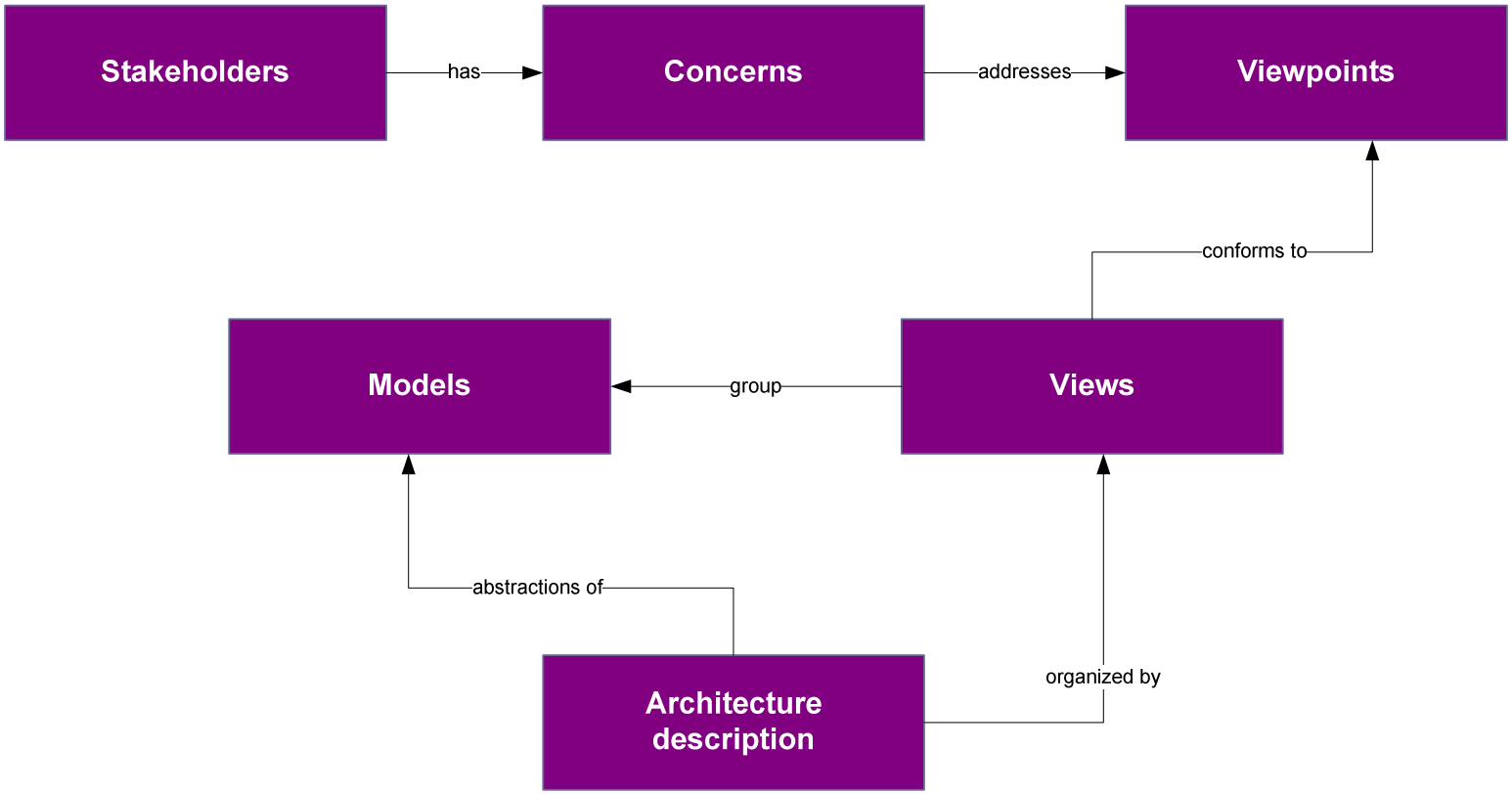


Figure 1—Conceptual model of architectural description

Meta-modelo Simplificado

La arquitectura se representa en un conjunto de vistas, de acuerdo a los concerns de los stakeholders



Elementos que forman parte de la comunicación

- **Diagramas de Contexto**
  - Define el scope de la aplicación
- **ViewPoint** (Perspectiva)
  - Un viewpoint determina los lenguajes (anotaciones, modelos, etc) que se usaran para describir la **view**.
- **View** (View)
  - Es la representación de una arquitectura con respecto a un **viewpoint** particular, se constituyen de View Packages y Modelos
- **View Packages & Models**
  - Son diferentes modelos o elementos utilizados para documentar una vista.
- **Escenarios**
  - Que muestran las interacciones de diferentes vistas
- **Decisiones de Arquitectura**

Agenda

#	Tema
1	Concepto de Comunicación y Entendimiento de Arquitectura
2	Frameworks de Arquitectura
3	ADL
4	Guía / Metodología de Comunicación
5	Conclusión

Frameworks de Arquitectura

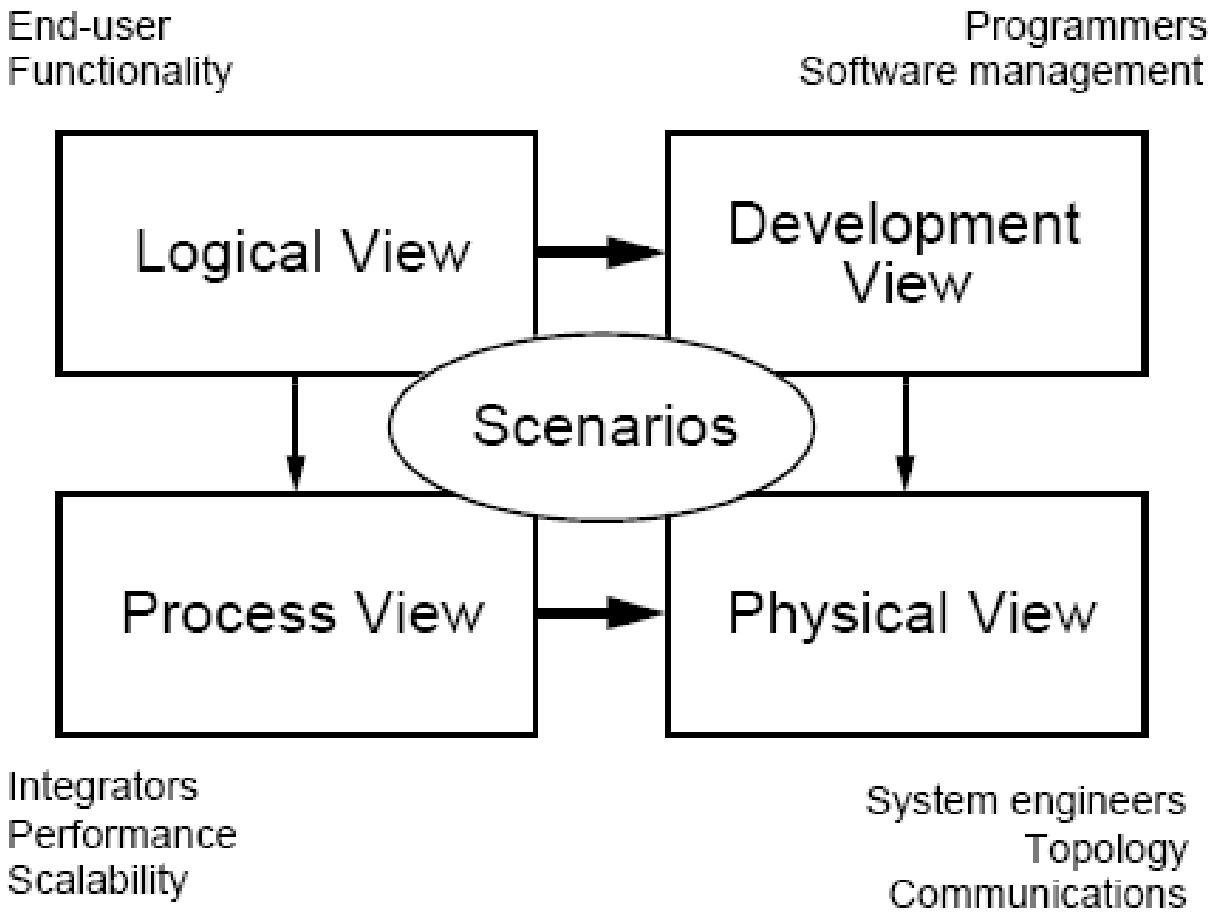
Los **frameworks de arquitectura de software** especifican las perspectivas (viewpoints) y relaciones necesarias, entre estas, para crear una arquitectura de software para sistemas específicos.

No solo sirven para comunicar sino tambien para todo el ciclo de arquitectura (analizar, diseñar y evaluar)

Frameworks de Arquitectura – 4+1 View Model…

- Este framework aplica a las aplicaciones enterprise y específicamente a las que utilizan tecnología orientada a objetos.
- Fue creada por Kutchen, empresa Rational, previo a la creación del RUP, como un framework para poder modelar la arquitectura y definir diferentes **viewpoint** para cada tipo de stakeholders.

Frameworks de Arquitectura – 4+1 View Model…



Frameworks de Arquitectura – 4+1 View Model…

Viewpoint	Logical View
Stakeholders	▪Usuarios Finales ▪Desarrolladores
Descripción	Es un viewpoint que representa los requerimientos funcionales. Es independiente de plataforma, por lo general representan el concepto del dominio del problema, o sea los objetos del negocio. Esta vista debería mapear los requerimientos con las clases y sus relaciones.

Frameworks de Arquitectura – 4+1 View Model…

Viewpoint	Process View
Stakeholders	▪Integradores ▪Desarrolladores ▪Grupo de Tecnología e Infraestructura
Descripción	Es un viewpoint que representa el modelo de procesamiento del sistema. Tiene en cuenta los atributos no funcionales como performance y availability.

Frameworks de Arquitectura – 4+1 View Model…

Viewpoint	Development View
Stakeholders	▪SCM Group ▪Build Team
Descripción	Es un viewpoint que representa la organización de los subsistemas y cantidad layers, interfases entre estas y dependencias.

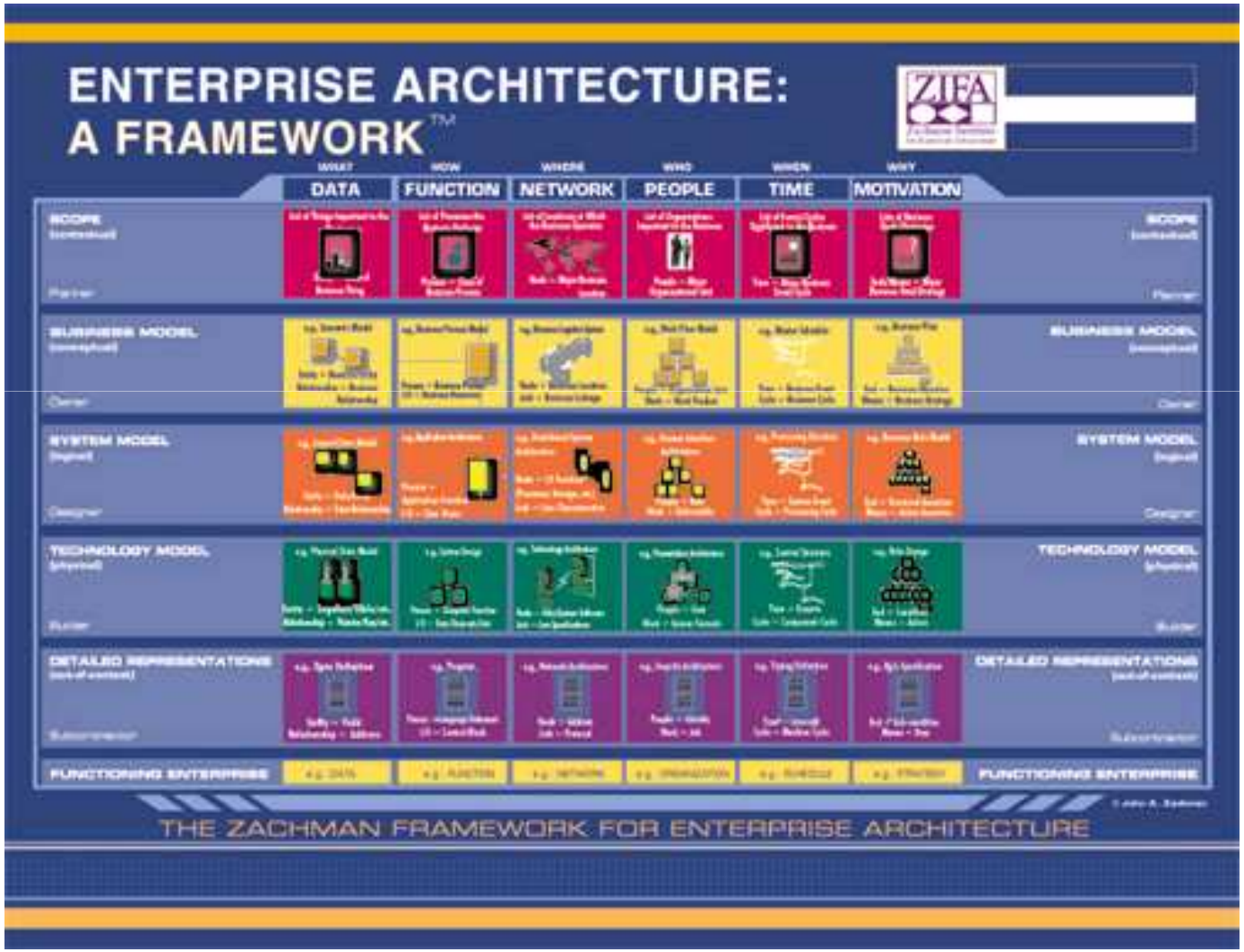
Frameworks de Arquitectura – 4+1 View Model…

Viewpoint	Physical View
Stakeholders	▪Build Team ▪SE
Descripción	Es un viewpoint que representa el mapeo entre el software y el hardware, como asi también su distribución. Tiene en cuenta los atributos no funcionales como, availability, reliability, parformance y scalability.

Frameworks de Arquitectura – 4+1 View Model…

Viewpoint	Scenario View
Stakeholders	▪Usuarios Finales ▪Desarrolladores ▪Operatores ▪Testers
Descripción	Es un viewpoint integra los viewpoint (vistas) juntas en los casos de uso y escenarios de estos. Especificando los requerimiento funcionales.

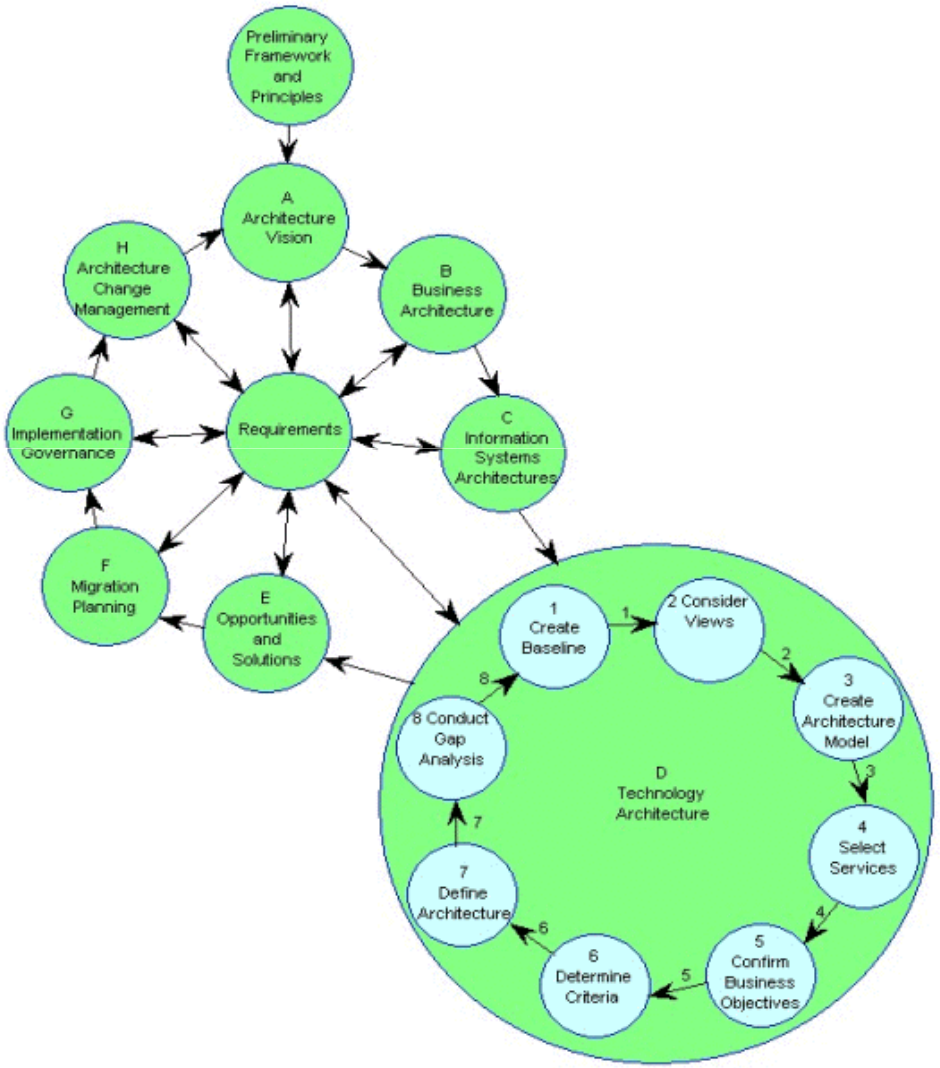
Frameworks de Arquitectura – Zachman Framework



Frameworks de Arquitectura – TOGAF

- El TOGAF (The Open Group Architecture Framework) es un framework para arquitecturas empresariales que provee un enfoque comprehensivo para el diseño, planificación, implementación, y gobierno de una arquitectura empresarial.
- La arquitectura esta típicamente modelada en cuatro niveles de dominio: Negocio, Aplicación, Datos, y Tecnología.
- Se proveen una serie de arquitecturas fundacionales para permitir al equipo de arquitectura preveer el estado actual y futuro de una arquitectura.

Frameworks de Arquitectura – TOGAF (Cont.)



Agenda

#	Tema
1	Concepto de Comunicación y Entendimiento de Arquitectura
2	Frameworks de Arquitectura
3	ADL
4	Guía / Metodología de Comunicación
5	Conclusión

ADL - Introducción

- Los **Architecture Description Language** (ADL) son un lenguaje formal usado para describir arquitecturas de software.
- Diferenciar un ADL de una notación no formal. No siempre un ADL es lo mejor.
- Existen diferentes ADLs, como
  - **Acme** (desarrollado por el CMU)
  - **AADL** (estandarizado por SAE)
  - **C2** (desarrollado por UCI)
  - **Darwin** (desarrollado por Imperial College London),
  - **Wright** (desarrollado por el CMU).
  - **UML** V2 ?????

Ejemplo de ADL - UML V2

- UML v2 (como otros ADL), tiene sus ventajas y sus desventajas. A continuación se hace una muestra del ejemplo y se termina con una conclusión acerca de la utilidad (o no) de este tipo de notación.

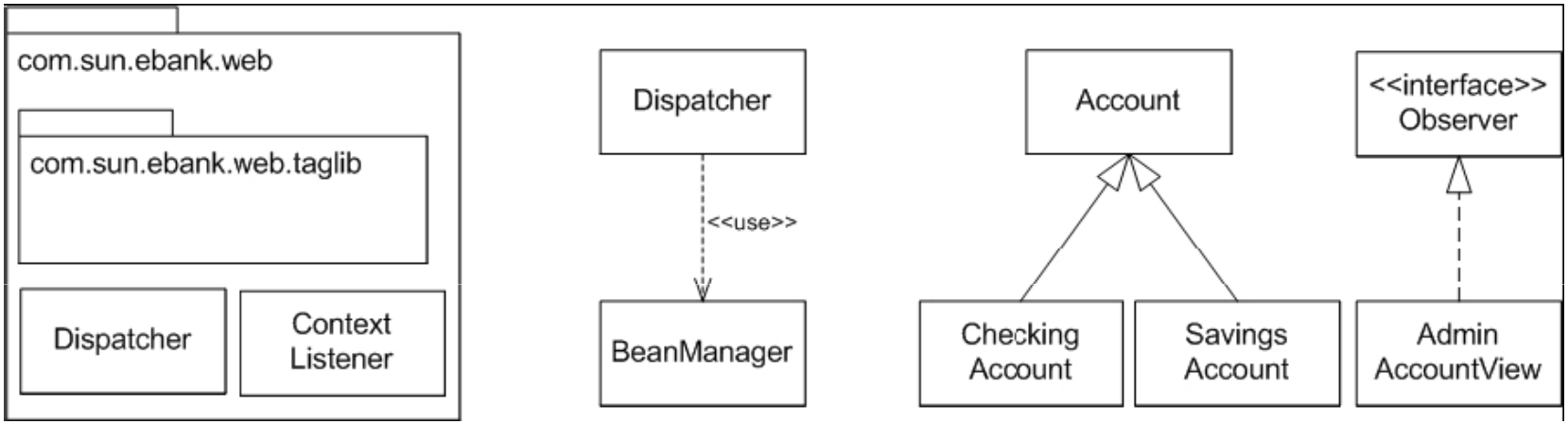
- UML v2 define una arquitectura Multi-vista
  - Module Views
  - Runtime views
  - Deployment views
  - Data Model

UML v2 – Module View

- Muestra la estructura de un sistema en terminos de unidades de implementación
- Elementos: modulos, ej., unidades de código que implementan cierta funcionalidad
- Relaciones:
  - A es parte de B: relación part-whole entre módulos
  - A depende de B: relación de dependencia entre módulos
  - A es un B: relación de especialización/genrealización entre módulos, o implementación de interfaces

UML v2 – Module View

UML Relations Between Modules



"Is part of"

Package contains subpackages or classes

"Depends on"

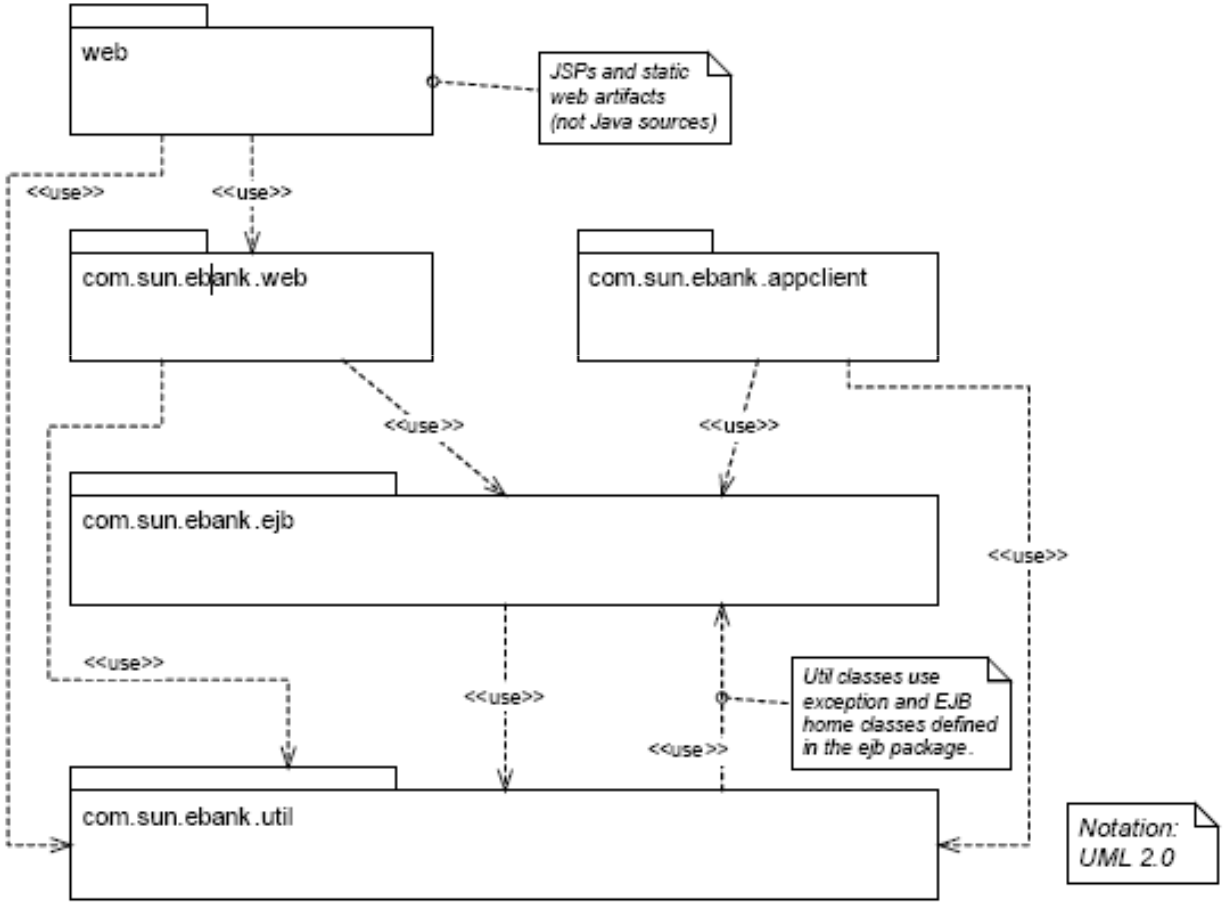
Dependency can be <<use>>, <<refine>>, <<instantiate>>, etc.

"Is a"

Generalization and interface realization

UML v2 – Module View

Module Usage Decomposition



UML v2 – Module View

Module View utilization

- Construcción
- Budgeting, work assignment, tracking
- Educación de nuevos desarrolladores
- Modificabilidad y análisis de impacto

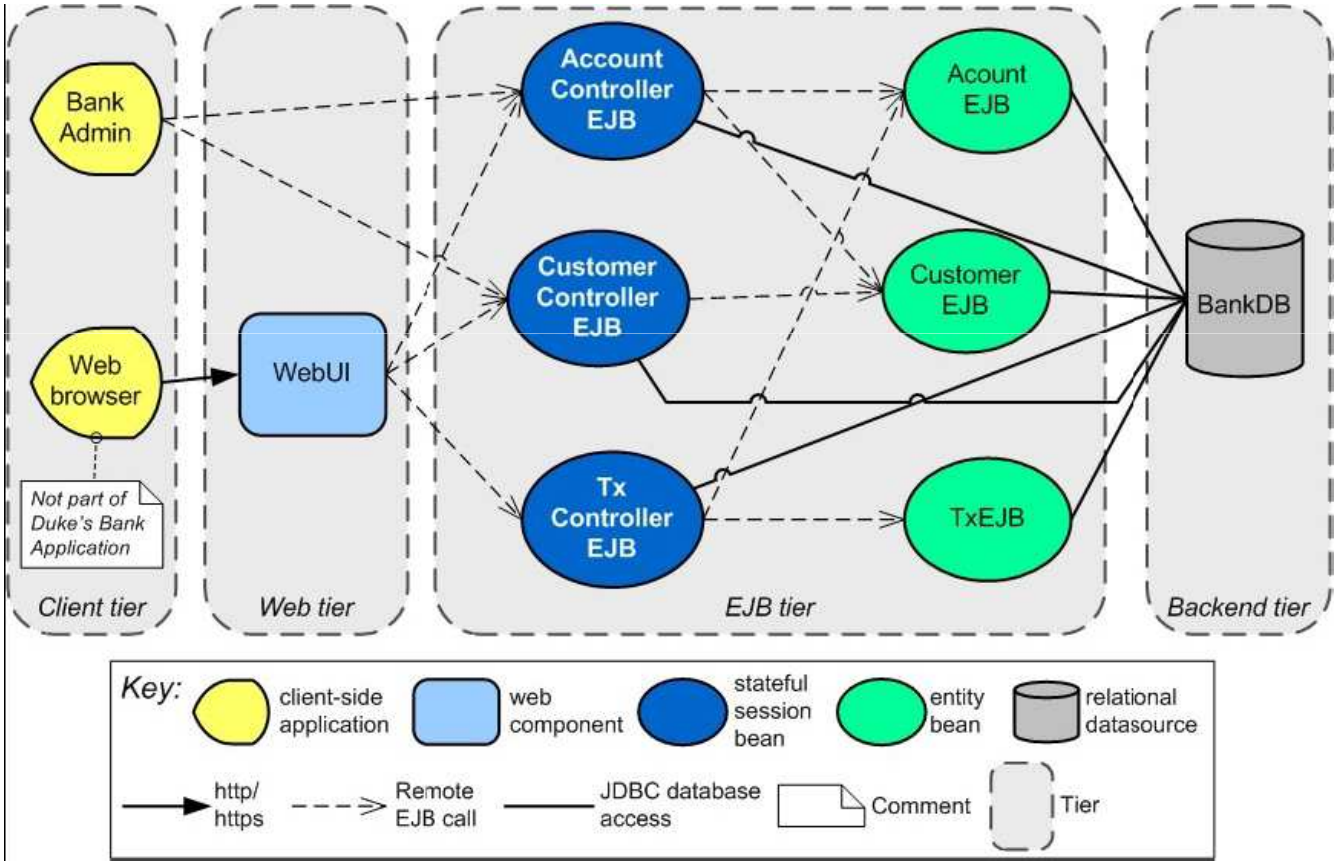
UML v2 – Runtime Views

Muestran la estructura de un sistema en tiempo de ejecución

- Elements:
  - Componentes que estan presentes en tiempo de ejecución (procesos,threads, EJBTM components, servlets, DLLs, objetos)
  - Repositorios de datos
- Relaciones:
  - Mecanismos de interacción basados en la tecnología
  - Se deben diferenciar:
    - Llamadas remotas y locales
    - Invocación síncrona y asíncrona

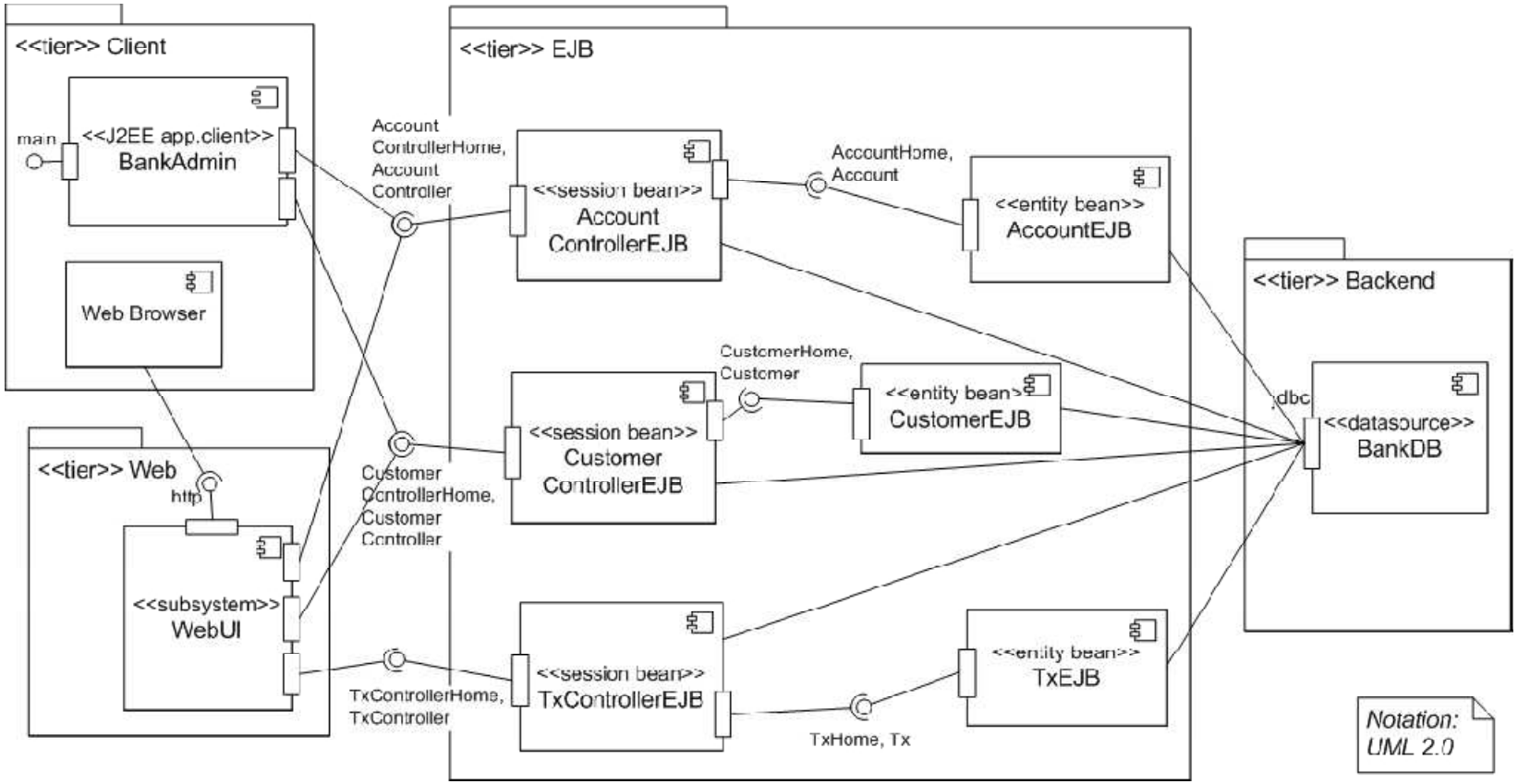
UML v2 – Runtime Views

Informal Notation



UML v2 – Runtime Views

Runtime Views – UML v2 Notation



UML v2 – Runtime Views

Runtime Views utiilization

- Explica:
  - Como interactuan los componentes en tiempo de ejecución
  - Que componentes se replican
  - Que componentes acceden a los repositorios de datos
- Educación—punto de entrada para saber como trabaja el sistema
- Analisis de propiedades de runtime
  - Performance
  - Security
  - Reliability

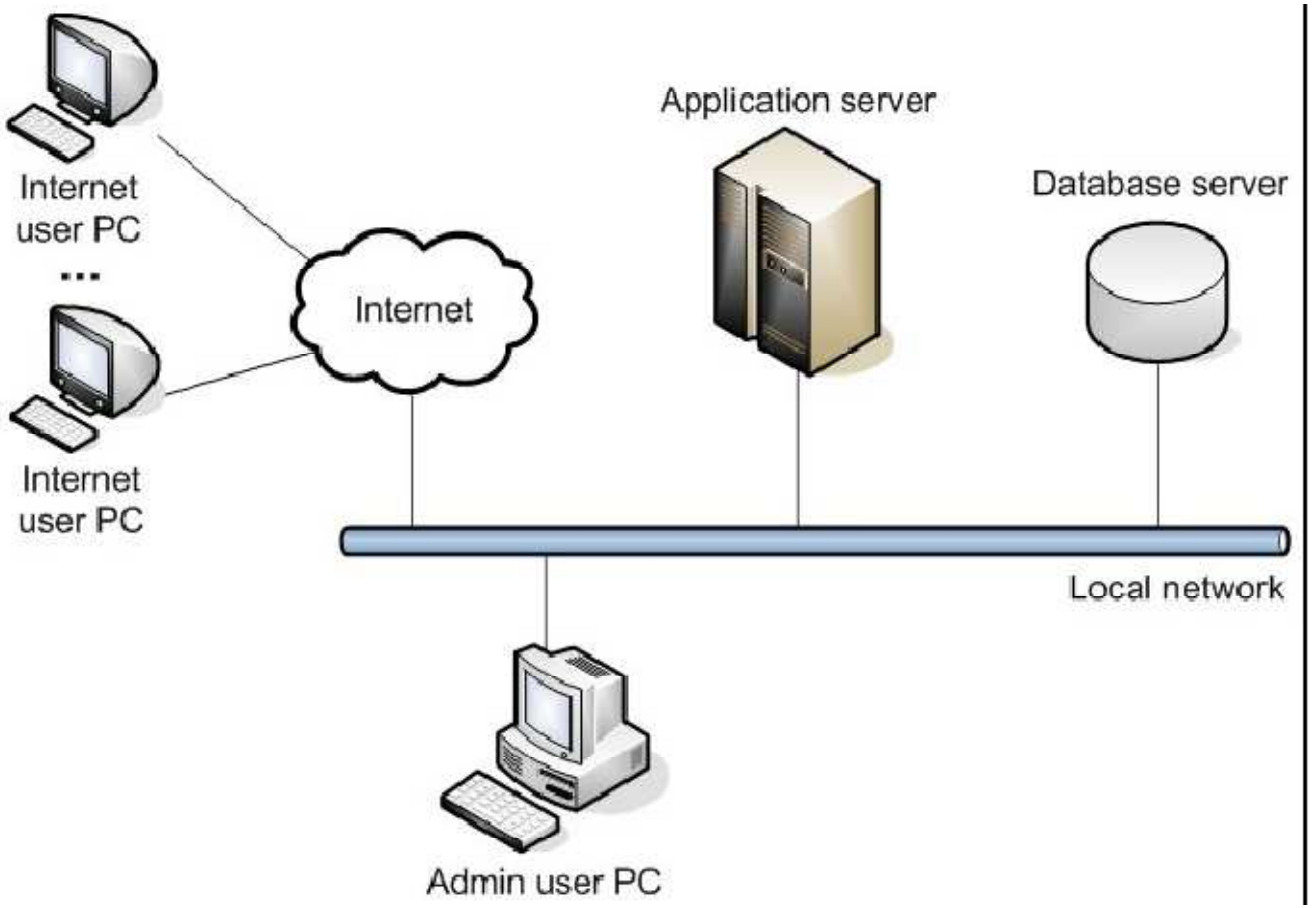
UML v2 – Deployment Views

Deployment Views

- Muestra al menos dos estructuras distintas pero relacionadas:
  - **Infraestructura Hardware** del ambiente productivo
  - La estructura de directorios y archivos del sistema implementado
- Elementos:
  - Nodos de comunicación y procesamiento (server machine, router)
  - Archivos, directorios
- Relations:
  - Mecanismos de interacción entre dos elementos son usualmente canales de comunicación
  - Contenedores: un directorio/archivo contiene otros directorios o archivos

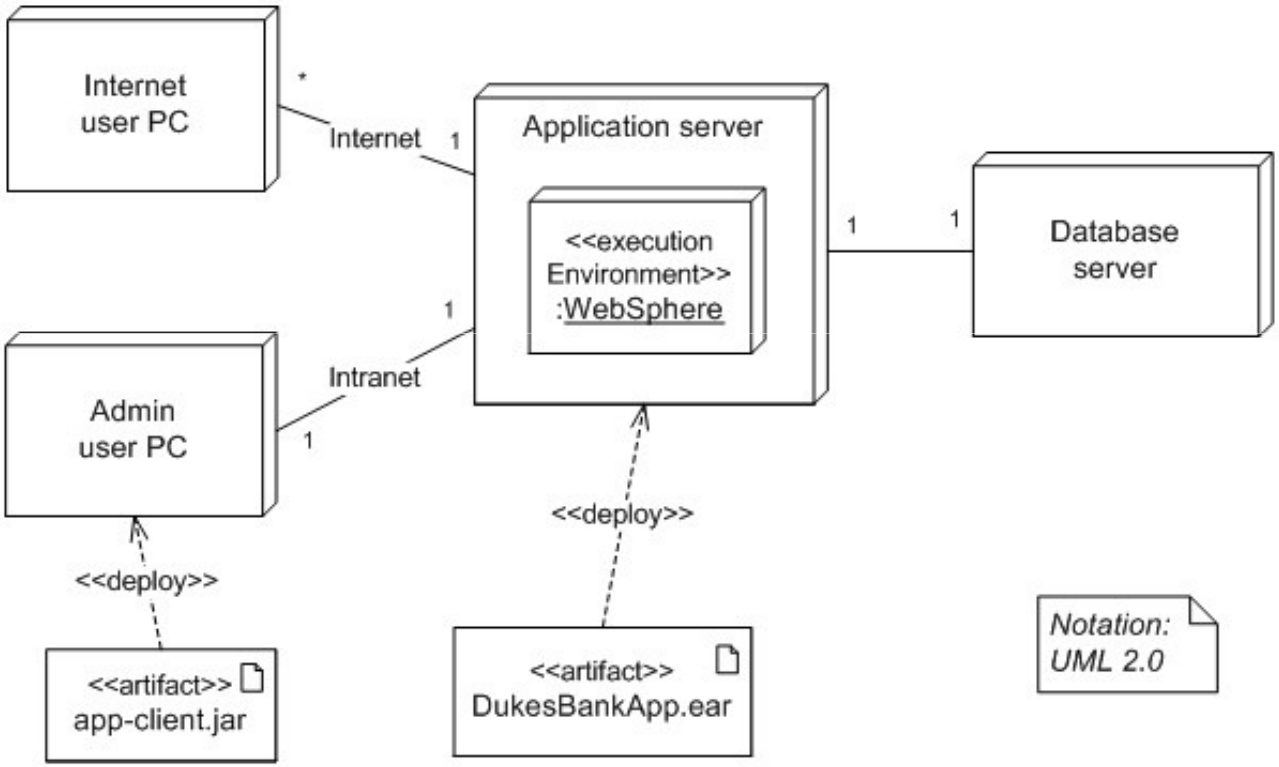
UML v2 – Deployment Views

Deployment Views – Informal Notation



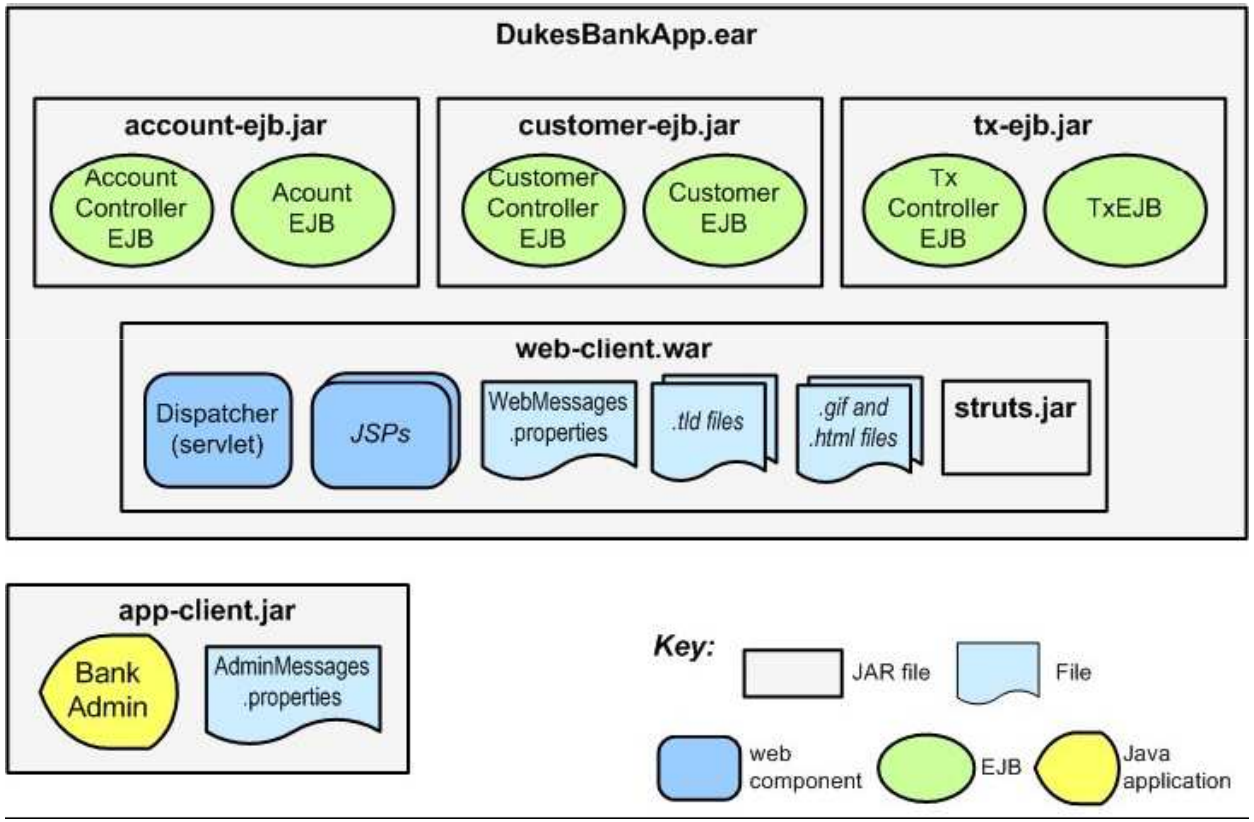
UML v2 – Deployment Views

Deployment Views – UML v2 Notation



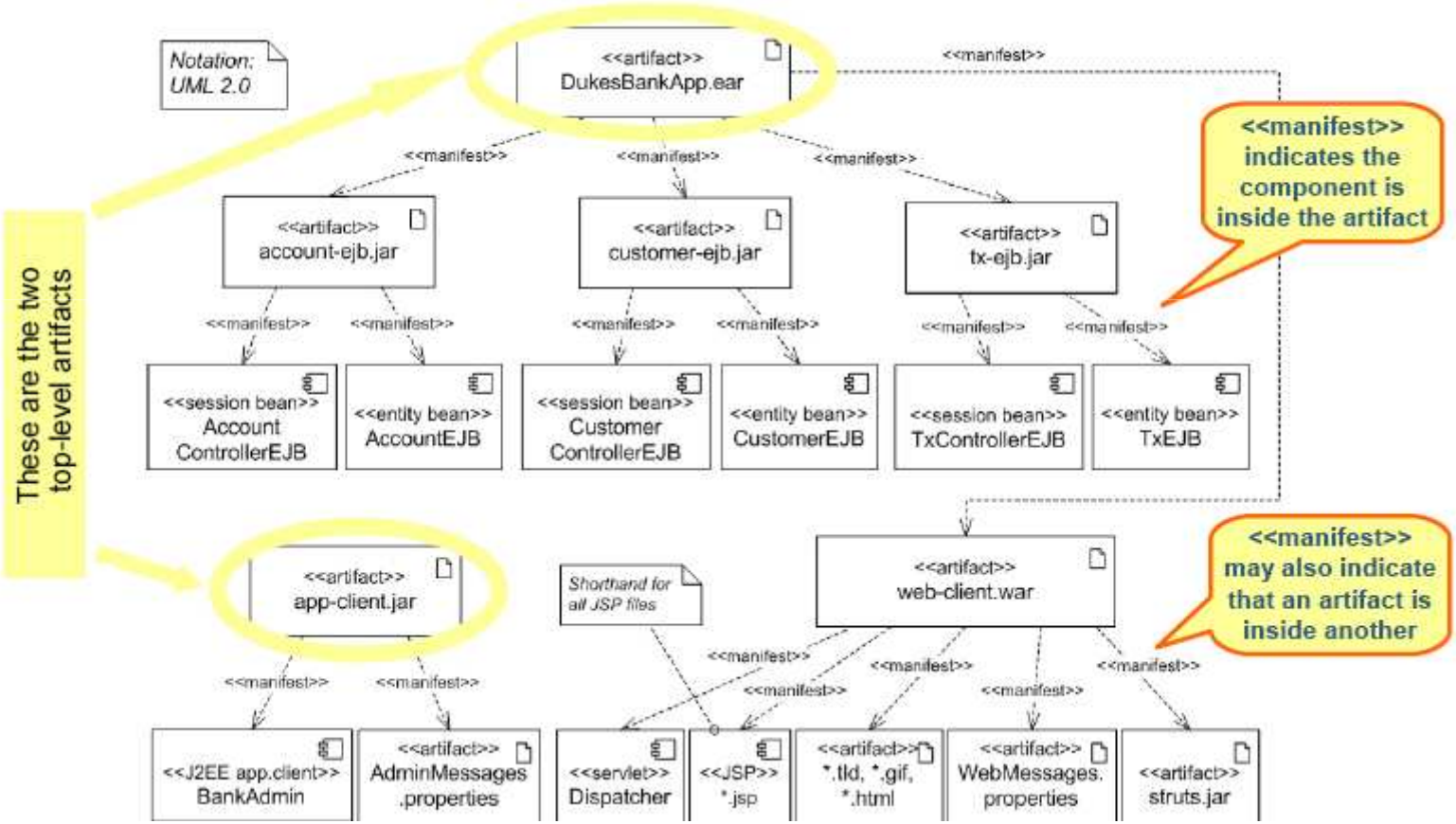
UML v2 – Deployment Views

Deployment Views – Informal Notation



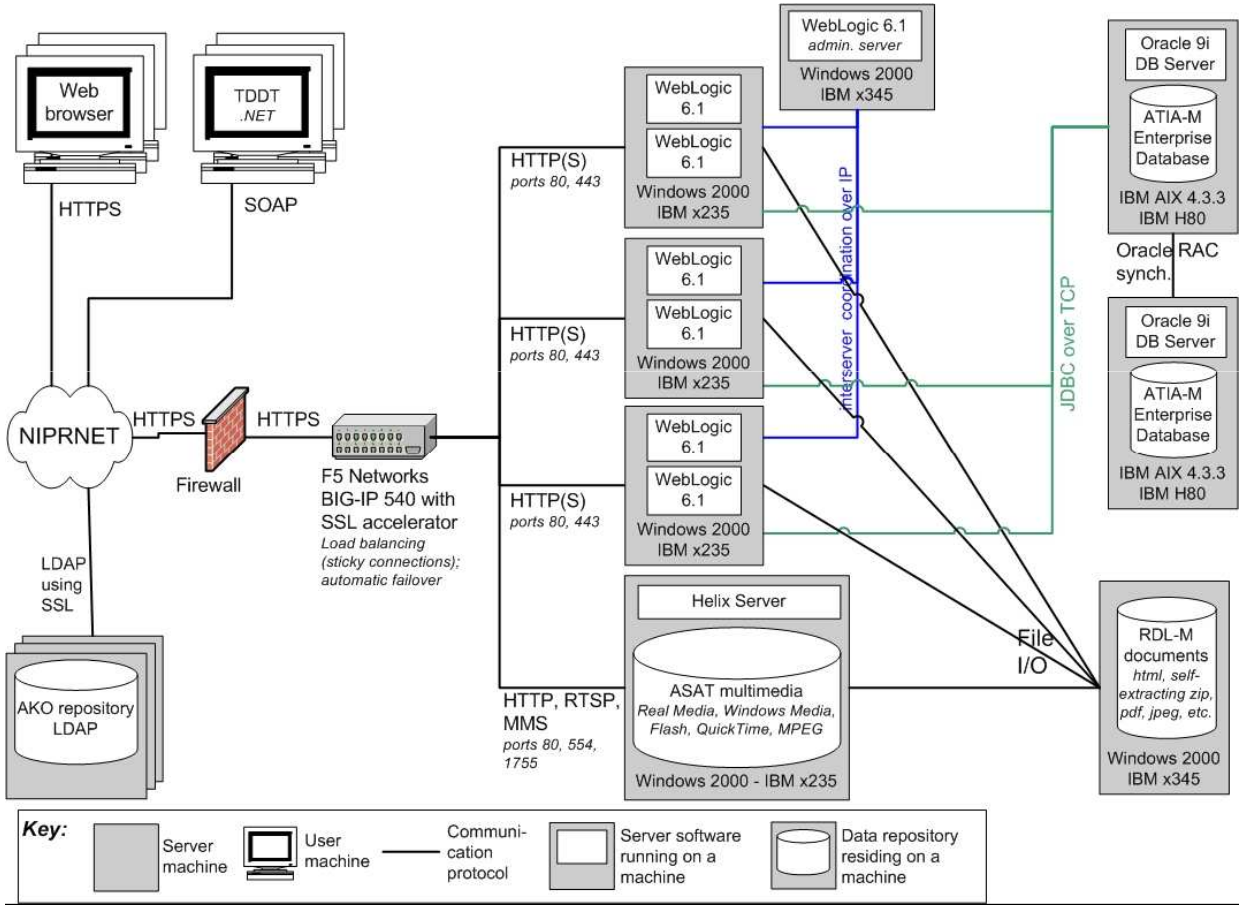
UML v2 – Deployment Views

Deployment Views – UML v2 Notation



UML v2 – Deployment Views

Deployment Views – Informal Notation



UML v2 – Deployment Views

Deployment Views utilization

- Definen procedimientos de implementación y operación
- Auditoría de fallas en tiempo de ejecución
- Planificación de opciones de compra
- Analiza:
  - Availability
  - Performance (throughput, bandwidth utilization)
  - Security
  - Reliability
- Educación y comunicación con stakeholder

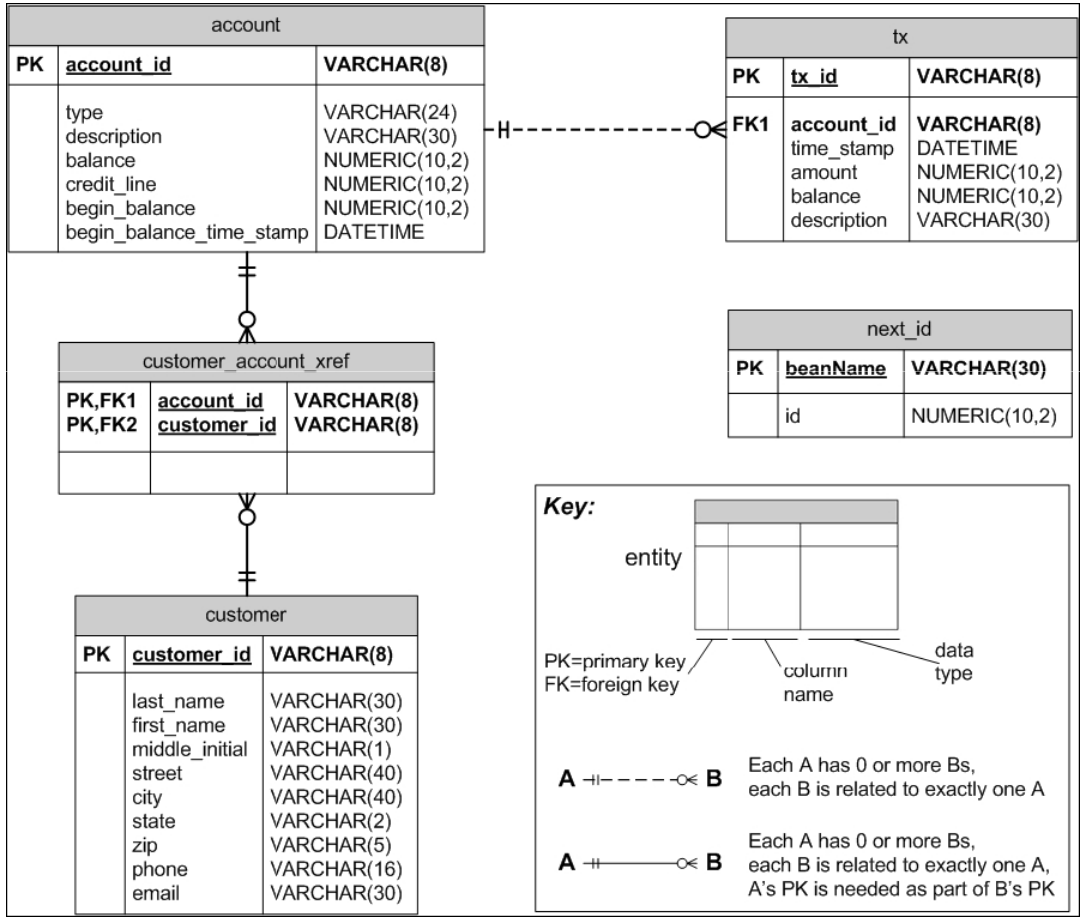
UML v2 – Data Model

Data Model

- Muestra la estructura del repositorio de datos
- Elementos: entidades (persisted domain elements)
- Relaciones:
  - Relaciones 1:1, 1:n y n:n
  - Generalización/specialización
  - Agregacion

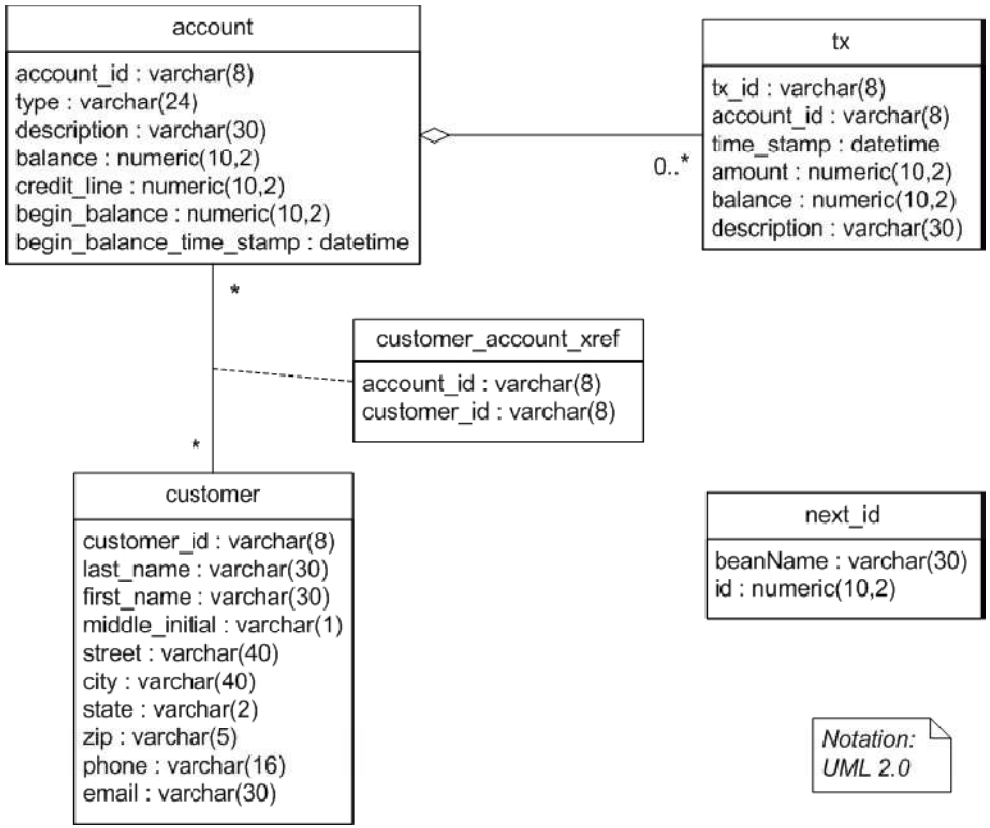
UML v2 – Data Model

Data Model – ERD Notation



UML v2 – Data Model

Data Model – UML v2 Notation



UML v2 – Data Model

Data Model utilization

- Blueprint para la base de datos física
- Referencia para todos los programas que manipulan datos
- Database tuning (normalization):
  - Para mejoras de performance and modifiabilidad
  - Para evitar redundancia
  - Para reforzar consistencia

UML v2 – Conlusión

- Se presenta a UML v2 como ejemplo de ADL y NO como una forma recomendable de hacerlo.
- Tanto el enfoque de UML v2, como de cualquier otro ADL, ninguno se va a adaptar al 100% de los casos, es decisión del arquitecto saber qué gráficos utilizar y en qué momentos.
- En determinados casos, incluso un gráfico informal (como los precedentes) pueden ser más útil que algún ADL.
- Como dijimos anteriormente, tenemos diferentes vistas para diferentes stakeholders, así mismo un stakeholder puede influenciar en el tipo de anotación a utilizar (ya sea formal, informal, ADL, etc.)

Agenda

#	Tema
1	Concepto de Comunicación y Entendimiento de Arquitectura
2	Frameworks de Arquitectura
3	ADL
4	Guía / Metodología de Comunicación
5	Conclusión

Guía para la comunicación de la Arquitectura

1. Diagrama de Contexto y Overview Diagram
2. Identificar Stakeholders y Concerns
3. Definir los ViewPoint necesitados
4. Documentar y Comunicar las Vistas
5. Documentar y Comunicar los Escenarios
6. Documentar las decisiones (Constantemente)
7. Armado de SAD (Wiki, Documento de Texto, etc)

1-Diagrama de Contexto y Arquitectura

- Es un diagrama esquemático que representa las ideas prestablecidas y los módulos/componentes candidatos de un sistema o arquitectura. Provee un resumen de los elementos conceptuales y sus relaciones en una arquitectura.
- Toda solución técnica debe tener al menos los siguientes elementos:
  - Actores o Roles Principales
  - Los módulos/componentes principales
  - Los Nodos principales
  - Repositorios de Datos
  - Como fluye la información
  - Las zonas de red

2-Identificar Stakeholders y Concerns

1. Listar los stakeholder (input del PM)
   - Team de desarrollo internos, Team de desarrollo de otras empresas, Sponsors, Usuario final, Consultores, Infraestructura, Enterprise Architect, etc
2. Rankearlos en que tan importantes son para el arquitecto de acuerdo a la dependencia y el poder.
3. **Etiquetar** los stakeholder con mayor alta dependencia.
4. Volver a ranckearlos, solos los de alta dependencia, teniendo la posición de acuerdo al alineamiento de objetivos y calidad en la relación,
5. **Etiquetar** los stakeholders con mala relación.
6. Para los etiquetados en el paso 3, establecer **buenos canales de comunicación** y que haya mucho feedback. Son importantes como input
7. Para los etiquetados en el paso 5, buscar **consenso constante** luego de cada versión de la arquitectura.

3-Definir los ViewPoint necesitados

- De acuerdo a los concerns de los stakeholders necesitados se definen o seleccionan los ViewPoints. Tener en cuenta que tipo de información necesita cada stakeholder:
  - Información detallada
  - Algunos detalles
  - Resumen
  - Nada
- Se pueden seleccionar de a algún framework de arquitectura
- Se pueden combinar de deferentes tipos y se pueden definir nuevos
- Identificar los canales de comunicación preferidos (la PNL puede ayudar muchísimo)
- Generalmente se definen sus elementos, tipos de relación, propiedades y restricciones

4-Documentar y Comunicar las Vistas

- Cada vista corresponde con un solo viewpoint
- Cada vista debe incluir:
  1. Las representaciones necesarias para mostrar la arquitectura (graficos o modelos) siguiendo el lenguaje definido en el viewpoint
  2. Catalogo de Elemento por Modelo
  3. Relación con el Diagrama de Contexto
  4. Glosario interno
  5. Comportamientos
  6. Interfaces

5-Documentar y Comunicar los Escenarios

- Se documentan escenarios mapeando diferentes vistas
- Es el mismo concepto de la 4+1
- Tener muy en cuenta a los stakeholder de alta dependencia.

6-Documentar las decisiones

- Es el lugar en donde se documentan todas las decisiones arquitectónicas a lo largo del ciclo de vida del proyecto.
- Cada decisión debe contar de los siguientes elementos:
  - Requerimiento/s, Atributo/s de Calidad o Restricciones directamente relacionado
  - Otras influencias y supocisiones (indirectamente relacionada)
  - Módulos y/o Componentes afectados
  - Deben tener un identificador
  - Alternativas evaluadas
  - Justificadación de la decisión tomada e impacto
  - Estado de la decisión
  - Decisiones relacionadas
- Un Wiki puede ayudar muchisimo a mantenerlas (ojo con los accesos)

7- Armado de SAD (Wiki, Documento de Texto, etc)

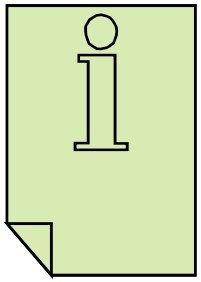
- Stakeholder
- Diagrama de Contexto
- Definiciones de Viewpoints
- Architecture Background
  - Problem Background
  - Solution Background
  - Architectural Drivers
- Views
  - Vista 1 (Representaciones, Elementos, Comprotamientos, Interfaces, etc)
  - Vista N
- Escenarios
- Decisiones Arquitectónicas (puede estar separado)
- Glosario

Agenda

#	Tema
1	Concepto de Comunicación y Entendimiento de Arquitectura
2	Frameworks de Arquitectura
3	ADL
4	Guía / Metodología de Comunicación
5	Conclusión

Conclusión

- Comunicar en general no es una tarea trivial, por que sería diferente para la Arquitectura de Software?
- Debe ser lo suficientemente abstracta para ser rápidamente entendida pero debe ser lo suficientemente detallada para el su analisis e implementación.
- Debe poder satisfacer las diferentes necesidades de los stakeholders, en un solo documento fácil de entender y navegar. Cada uno le da un uso de acuerdo a sus necesidades
- La creatividad para comunicar la arquitectura es la guia fundamental para el éxito, es más importante lograr el entendimiento de las decisiones que cumplir con una notación formal que dificilmente lleguen a tener todo lo necesario.



Referencia

- Software Architecture in Practice, Second Edition. Len Bass, Paul Clements, Rick Kazman,. Addison Wesley, 2003, ISBN 0-321-15495-9.
- Documenting Software Architectures: Views and Beyond. Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford. Addison Wesley, 2002, ISBN 0-201-70482-X.
- IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Std 1471-2000, September 2000.
- ACME http://www.cs.cmu.edu/~acme/
- The Open Group http://www.opengroup.org/
- Zachman Framework http://www.zifa.com/